



Faculty of Information and Communication Technology

**ALGORITHM-PROGRAM VISUALIZATION MODEL:
AN INTEGRATED SOFTWARE VISUALIZATION TO SUPPORT
NOVICES' PROGRAMMING COMPREHENSION**

Affandy

Doctor of Philosophy

2015

**ALGORITHM-PROGRAM VISUALIZATION MODEL:
AN INTEGRATED SOFTWARE VISUALIZATION TO SUPPORT
NOVICE'S PROGRAMMING COMPREHENSION**

AFFANDY

**A thesis submitted
in fulfilment of the requirements for the degree of Doctor of Philosophy**

Faculty of Information and Communication Technology

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

2015

DECLARATION

I declare that this thesis entitled “Algorithm-Program Visualization Model: An Integrated Software Visualization to Support Novices’ Programming Comprehension“ is the result of my own research except as cited in the references. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.

Signature :

Name : Affandy

Date :

APPROVAL

I hereby declare that I have read this thesis and in my opinion this thesis is sufficient in terms of scope and quality for the award of Doctor of Philosophy

Signature :

Supervisor Name : Prof. Dr. Nanna Suryana Herman

Date :

DEDICATION

In the name of Allah, the most beneficent, the most merciful

Al-hamdulillahi rabbil 'alamin

All the praises and thanks be to Allah, the Lord of the 'Alamin

this work is dedicated to:

my beloved parents

[H. Achmad Rozy & almarhumah Hj. Djumirah]

my endless loves, my wife and my daughter

[Trabalista Purwaningrum & Nobelia Salma Aisyahfira]

my siblings and the families

[Heni Isnaeni, Iin Indriani, and Roziqin]

my parents-in-law and the families

[Heru Sutrisno & Mamiek, Yulianto, Nuning, and Tatik]

ABSTRACT

Computer programming is the essential foundation for the other basic skills in Information Technology knowledge areas. Success in this field requires complex knowledge and skill. Mostly, conventional programming courses have been delivered based on the programming textbooks with professional developer tools which focus on the syntax or semantic through the coding task. The role of Software Visualization (SV) has been involved to overcome the complexity and problems in the learning programming. It represents the abstractness of the program in graphical views or illustrations of its entities. Nevertheless, the outcome of the learning still remains poor. Through multi-methodological approach, this research aimed to improve the effectiveness of the visualization as the program comprehension tool. It is found that the interrelated tasks in the programming process, with its various abstractions, and timing in delivering the feedback, need to be addressed with the equal attention in learning to program. Taking into account from those main issues, this study introduces the new model of integrated algorithm-program visualization (ALPROV) for developing program comprehension tool. This model is then to be used in the prototype tool development that is called 3De-ALPROV (Design Development Debug – Algorithm Program Visualization). The efficacy evaluation of the prototype is based on pre- and post- test of the students' programming performance. The programming performances from the treatment and control group are compared to analyze the effect of using the proposed tool in learning programming. Respondents are first-year bachelor students who lack of programming knowledge and experience. Analysis proved that using the program comprehension tool, which has been developed using integrated ALPROV model significantly improved the treatment group's programming performance. Conducting other experiments as the extended study, such as seek for a larger group of respondents, conduct the experiments throughout the necessary period, and use various methods for programming assessment and analysis may improve the findings of this research.

ABSTRAK

Pengaturcaraan komputer adalah asas penting bagi kemahiran yang lain dalam bidang teknologi maklumat. Kejayaan di bidang ini memerlukan pengetahuan dan kemahiran yang kompleks. Kebanyakan kursus pengaturcaraan konvensional disampaikan berdasarkan buku teks pengaturcaraan dengan alat pembangun profesional yang tertumpu kepada sintak/semantik melalui tugas penulisan program. Visualisasi Perisian (SV) telah dibabitkan bagi mengatasi kerumitan dan masalah dalam pembelajaran pengaturcaraan. Ianya mendedahkan keabstrakan program dalam bentuk grafik atau ilustrasi dari entitinya. Walau bagaimanapun, masih terdapat kekurangan dalam hasil pembelajaran. Melalui pendekatan multi-methodological, matlamat kajian ini untuk meningkatkan keberkesanan visualisasi sebagai alat pemahaman program. Kajian ini mendapati bahawa tugas-tugas yang saling berkait dalam proses pengaturcaraan dengan pelbagai abstraksi program dan penyampaian maklum balas, perlu ditangani dengan perhatian yang sama masa belajar membuat program. Mengambil kira dari isu utama ini, kajian ini memperkenalkan model baru dari algoritma-program visualisasi bersepadu (ALPROV) untuk membangunkan alat pemahaman program. Model ini digunakan dalam pembangunan prototaip yang dipanggil dengan nama 3DE-ALPROV (Rekabentuk Pembangunan Debug - Algoritma Program Visualisasi). Penilaian keberkesanan prototaip adalah berasaskan kepada peperiksaan pra dan pasca pencapaian pengaturcaraan pelajar. Pencapaian pengaturcaraan daripada kumpulan perlakuan dan kawalan, dibandingkan dengan memerhati, mengukur, dan menganalisis keberkesanan penggunaan alat visualisasi yang dicadangkan dalam mempelajari pengaturcaraan. Responden adalah pelajar sarjana muda tahun pertama yang kurang pengetahuan dan pengalaman dalam pengaturcaraan. Analisis membuktikan bahawa menggunakan alat pemahaman pengaturcaraan yang telah dibangunkan dengan menggunakan model ALPROV bersepadu dapat meningkatkan pencapaian pengaturcaraan dari pada kumpulan perlakuan dengan signifikan. Menjalankan eksperimen yang lain sebagai kajian lanjutan, iaitu mendapatkan kumpulan responden yang lebih besar, eksperimen dalam tempoh yang cukup, menggunakan pelbagai kaedah untuk penilaian pengaturcaraan dan analisis boleh menambah baik kepada hasil kajian ini.

ACKNOWLEDGEMENTS

It is a privilege and a great pleasure to thank to many people who made this thesis possible.

First and foremost I would like to express my gratitude and sincere appreciation to my supervisor, Professor Dr. Nanna Suryana, through close interaction for support, encouragement, guidance and critics that continuously influence my way of thinking about research and living the life. The gratitude goes also to my co-supervisor, Associate Professor Dr. Burairah Husin, for his endless source of energy for sharing knowledge, expertise and experiences.

I would like to extend my special appreciation to Dr. Ir. Edi Noersasongko that has provided this opportunity, resources and supports for making it happen. To Dr Abdul Syukur for encourage, support, and motivate me to pursue my PhD. And to all staffs from the Faculty of Information and Communication Technology and the International Office in UTeM and also in UDinus for providing the excellent services and a comfortable environment during my study.

My sincere appreciation also extends to the evaluation committee, Professor Dr. Baharuddin Aris, Dr. Gede Pramudya Ananta, and Assoc. Professor Dr. Burhanuddin Mohd. Aboobaidar, for their valuable input and guidance, and to all of my colleagues who have provided assistance at various occasions especially to Dr. Fahmi Arif, Romi Satrio, Basem Zughoul, Daniel Hartono, and Mohamed Doheir.

Heartfelt thanks to my beloved parents, and parents-in-law, my brother, and my sisters for supports, prayer, and understanding. And last but not least for the most precious in my life, my wife Trabalista, and my daughter Nobelia; for your immense support I shall remain forever grateful since without both of you it would all be meaningless.

May the Almighty Allah will always guide you in every step you take, bless you with more than enough, and make things easy for you when life gets rough.

TABLE OF CONTENTS

	PAGE
DECLARATION	
DEDICATION	
ABSTRACT	i
ABSTRAK	ii
ACKNOWLEDGMENTS	iii
TABLE OF CONTENTS	iv
LIST OF TABLES	ix
LIST OF FIGURES	xi
LIST OF APPENDICES	xiv
LIST OF ABBREVIATIONS AND GLOSSARIES	xvi
LIST OF PUBLICATIONS	xix
CHAPTER	1
1. INTRODUCTION	1
1.1 Research Background	1
1.1.1 The Tasks of Programming	5
1.1.2 The Multiple Representations of the Program	8
1.1.3 The Role of Visualization in Programming Comprehension	11
1.1.4 The Difficulties of Learning Programming / CS1	15
1.1.5 The Paradox of Software Visualization	17
1.2 Challenges in Developing Integrated Visualization Model for Novice Students	21
1.3 Problem Statement	23
1.4 Research Objectives	28
1.5 Scope of Research	30
1.6 Research Area	31
1.7 Research Contribution	34
1.8 Thesis Organization	35

1.9	Summary	37
2.	CONCEPTUAL FRAMEWORK	39
2.1	Introduction	39
2.2	Computer Programming	40
2.2.1	What is Programming?	40
2.2.2	Programming-related Tasks.	45
2.2.3	Programming Comprehension Model	56
2.2.4	The Common Threads of Program Comprehension Theories.	61
2.3	Visualization	65
2.4	Software Visualization (SV)	68
2.4.1	Taxonomy of Software Visualization	70
2.4.2	State of the Art of Software Visualization Systems	77
2.5	Summary of Current Software Visualization Systems	91
2.5.1	Issue on Multiple Programming Tasks.	91
2.5.2	Issue on Multiple Representations of the Program.	92
2.5.3	Issue on Language Selection	95
2.5.4	Issue on the Edit-Run Environment and Execution Time	95
3.	RESEARCH METHODOLOGY	97
3.1	Introduction	97
3.2	Research Design and Procedure	97
3.2.1	The Construction of Theory	98
3.2.2	Survey Studies	99
3.2.3	Experimentation	100
3.2.4	System Development	100
3.3	The Operational Framework	101
3.4	Preliminary Survey	103
3.4.1	Respondents	104
3.4.2	Survey Questions and Tasks	105
3.4.3	Procedures	107
3.4.4	Preliminary Survey Findings	107
3.5	Analysis of Preliminary Survey Findings	113

3.6	Summary	118
4.	THE DEVELOPMENT OF INTEGRATED ALGORITHM-PROGRAM VISUALIZATION MODEL	120
4.1	Introduction	120
4.2	Students Need A Better Approach to Learn About Programming.	120
4.3	Requirements for Integrated Program Comprehension Tool Model.	123
4.3.1	Requirements Based on Program Comprehension Theories	123
4.3.2	Requirements Based on Own Practical Experiences and Observations.	129
4.4	The Integrated Algorithm-Program Visualization Model	133
4.5	Model of Programming-Process Support in Visualization Tool	136
4.5.1	The Initial Steps.	139
4.5.2	The Iterative / Progressive Steps	144
4.6	Model of Multiple Representation of Program.	156
4.6.1	The Higher-level of Program Representation	159
4.6.2	The Intermediate-level of Program Representation	161
4.6.3	The Lower-level of Program Representation	163
4.7	Semantic Feedback.	166
5.	PROTOTYPE DEVELOPMENT AND IMPLEMENTATION OF THE INTEGRATED ALGORITHM PROGRAM VISUALIZATION (ALPROV) TOOL	174
5.1	Introduction	174
5.2	Rapid Prototyping Development and Evaluation	174
5.3	Requirement Analysis	181
5.4	Design of the Prototype	186
5.4.1	Flowchart	186
5.4.2	Prototype Architecture	188
5.4.3	List of Components	192
5.4.4	Design of the Workflow Modelling	194
5.4.5	Activity for Constructing Algorithm Design	195

5.4.6	Activity for Comprehending an Existing Program or Algorithm Design	196
5.4.7	Activity for Controlling the Semantic Feedback	197
5.4.8	Activity for Managing Files	199
5.5	Prototype of 3De-ALPROV Program Comprehension Tool	200
5.5.1	User Interface of 3De-ALPROV	200
5.5.2	Using 3De-ALPROV in Learning Introductory Programming	202
5.6	Implementation of 3De-ALPROV in an Introductory Programming Course.	206
5.6.1	Respondents	207
5.6.2	Survey Questions	210
5.6.3	Procedures	212
5.6.4	Descriptive Statistics Results	215
5.7	Summary	229
6.	RESULT AND DISCUSSION	231
6.1	Introduction	231
6.2	Formulated Research Questions and Hypothesis	231
6.3	Proposed Solution for the Less Attention of Learning Programming as a Process (Research Problem 1)	233
6.3.1	Programming-related Tasks	234
6.3.2	Model of Programming Process Support in the Program Comprehension Tool	236
6.4	Proposed Solution for the Inattention of the Multiple Program Abstraction (Research Problem 2)	239
6.4.1	Multiple Representations Model of the Program Abstractions.	240
6.4.2	Adaptive Syntax/Semantic Feedback Model in the Program Comprehension Tool	245
6.5	The Integrated Algorithm-Program Visualization Tool Model	248
6.6	Proposed Solution for the Lack of the Integrated Program Comprehension Tool (Research Problem 3)	250
6.6.1	The 3De-ALPROV as the Prototype of the Integrated Program Comprehension Tool	250

6.6.2	The Effectiveness of the Integrated Program Comprehension Tool	252
6.7	Summary	258
7.	CONCLUSION AND FUTURE WORKS	260
7.1	Summary of Research	260
7.2	Research Conclusions	261
7.2.1	Conclusion of the Identification of Programming-related Tasks (RO#1)	261
7.2.2	Conclusion Related to Multiple Representations of the Program Abstraction (RO#2)	263
7.2.3	Conclusion of the Proposed Adaptive Feedback for The Visualization Tool (RO#3)	264
7.2.4	Conclusion Related to the Construction of Lab-Prototype Tool (RO#4)	266
7.2.5	Conclusion of the Effectiveness Study of the Proposed Model (RO#5)	267
7.3	Future Works	269
	REFERENCES	271
	APPENDICES	299

LIST OF TABLES

TABLE	TITLE	PAGE
1.1	Level and Program Abstraction of Current SV Tools	20
1.2	Research Problems and Related Literature Support	27
1.3	Relation between Research Problems, Questions and Objectives	29
2.1	Evolution on Computer Programming Definition	43
2.2	A Programming Framework (Robins et al., 2003)	48
2.3	Literatures Evolution in the Programming Tasks Definition	50
2.4	Various Studies of Program Comprehension Model	60
2.5	Taxonomy of Engagement (Naps et al., 2003a)	76
2.6	Taxonomy of Extended Engagement (Myller et al., 2009)	77
2.7	Comparison of Current Software Visualization Tools	93
3.1	Respondents of Preliminary Observations	104
3.2	Survey Instruments for Trace-code And Write-code Test	105
3.3	Preliminary Survey Result at UTeM	108
3.4	Preliminary Survey Result at UDinus	110
3.5	Descriptive Statistic of Test#2 from All Respondents	112
3.6	Comparison of Mean for the Second Trace-Code Test	112
3.7	Comparison of Mean for the Second Write-Code Test	113
4.1	Program Comprehension Tool Requirements based on Practical Experiences	132
4.2	Descriptions of Primary and Auxiliary Plans	146
4.3	Relationship of Programming-tasks and Program Representations	159
5.1	Expert-evaluation and Improvements of User Interface Design	178
5.2	Result of The Computer System Usability Questionnaire	179
5.3	Usability Factor's Score Based on The CSUQ's Results	180
5.4	Time Required for Prototype Development	181
5.5	Requirements of Integrated 3De-ALPROV Tool.	182
5.6	Requirement and Product Features Relationship	183

5.7	Flowchart Symbols in The Prototype Tool Based on IBM Flowcharting Template (x20-8020)	187
5.8	Summary of Demographic Information of Respondents	216
5.9	Statistic Analysis of Self-Efficacy in The Pre-Test	217
5.10	Statistic Analysis of Self-Efficacy in The Post-Test	218
5.11	Comparison of Self Efficacy Score Between Pre- and Post-Test	220
5.12	Comparison of Self Efficacy Score Between Control and Treatment Group	220
5.13	Descriptive Statistic of Programming Score in The Pre-Test	221
5.14	A Two-Sample Independent Difference Test Based on The Programming Pre-Test Score	222
5.15	Descriptive Statistic of Programming Score in The Post-Test	223
5.16	A Two-Sample Independent Difference Test Based on The Programming Post-Test Score	224
5.17	Difference Test between Pre-Test and Post-Test Score	226
5.18	Descriptive Statistics of Programming Score Based on Type of Question	227
5.19	Independent-Sample Test between Group Based on The Type of Questions	228
5.20	Dependent-Sample Test between Test Sessions Based on The Type of Questions	228

LIST OF FIGURES

FIGURE	TITLE	PAGE
1.1	Research on Teaching and Learning Introductory Computer Programming	4
1.2	Programming Process and Software Process	6
1.3	Structure of the Human Memory System (Atkinson, Shiffrin, 1968)	11
1.4	Price's Taxonomy of Software Visualization (Price et al., 1993)	13
1.5	Research Planning and Its Boundaries	33
2.1	Relationship Between Problem, Programming, and Solution	44
2.2	Programming Process as A Small-Scaled of Software Process	54
2.3	The Common Threads of Program Comprehension Theories	62
2.4	Various Types of Visualization from Different Fields	67
2.5	Timeline of Software Visualization Taxonomy	73
2.6	Alice 3D is running amusementPark (University, 2011)	79
2.7	ALVIS Live! Program Visualization Based on What You See Is What You Code (WYSIWYC) (Hundhausen, Brown, 2007)	80
2.8	Data Structure Visualization Demonstrates an En-queuing Process in an Array	82
2.9	Jeliot 3 with Contextual Pop-Up Question.	83
2.10	jGRASP present the multiple sychronized visualitations of TreeMap data structure (Hendrix et al., 2004)	85
2.11	MatrixPro Demonstrates a Binary Search Tree (Karavirta, 2009)	87
2.12	Flowchart Execution in RAPTOR Environment (Carlisle et al., 2005)	88
2.13	Representation Data in Memory Using The Teaching Machine	89
2.14	Step-by-step Execution Performed in ViLLE (Rajala et al., 2007)	90
3.1	A Multi-methodological Research (Jay F. Nunamaker et al., 1991)	98
3.2	Operational Framework of The Research and Its Related Chapters	103
3.3	Distribution of Surveys Respondents	105
3.4	Output Design for Write-Code Test	106
3.5	Score of Trace-Code Test From UTeM Respondents	109
3.6	Score of Write-Code Test From UTeM Respondents	109

3.7	Score of Trace-Code Test From UDinus Respondents	111
3.8	Score of Write-Code Test From UDinus Respondents	111
4.1	General Relation of Program Comprehension Model	124
4.2	Requirements of Comprehension Tools Based on Comprehension Model	126
4.3	The Integrated Algorithm-Program Visualization Model	134
4.4	Model of Programming-Process Support for Program Comprehension Tool.	138
4.5	Example of General Problem Description of GCD	140
4.6	Requirement Analysis Through Formulating the Problem	141
4.7	Solution Strategy	142
4.8	Goal Decomposition	142
4.9	Data and Resource Description	143
4.10	An Example of Solution Structure and Logical Data Flow Chart	145
4.11	Constructed Program Design Using Basic-Plan	148
4.12	Transformation of Program Abstraction in Multiple Representations	158
4.13	Source Code in Pretty-Printing Format	165
4.14	Visual Semantic-syntax Feedback	169
4.15	Relationship of Programming-tasks and Semantic Feedback	170
5.1	Development and Implementation of 3De-ALPROV Tool	175
5.2	Actor Generalization	184
5.3	Use Case Diagram of 3De-ALPROV	185
5.4	Plans in The 3De-ALPROV Tool	188
5.5	Architecture of Prototype System	189
5.6	Spatial Layout Design for Workspace	191
5.7	Class Diagram Relationships in Prototype Design	193
5.8	The Structure of User Interface in 3De-ALPROV	200
5.9	The User Interface of 3De-ALPROV	201
5.10	The 3De-ALPROV Menu	202
5.11	Defining a Data Structure in Variable Definition Plan.	203
5.12	Highlighted Variable and Code to Describe the Execution of Input Plan	204
5.13	The Current State while The If-Else Plan is Being Executed	204
5.14	The Final State of The Designed Algorithm	205
5.15	Specifying a Sample of Novice Students	207
5.16	Evaluation Procedure During 3De-ALPROV Tool Implementation	214
5.17	Chart of Respondents' Demographic Information	216
5.18	Histogram of Self Efficacy Score in The Pre-Test with Normal Curve	217
5.19	Histogram of Self Efficacy Score in The Post-Test with Normal Curve	219

5.20	Histogram of Programming Score in The Pre-Test with Normal Curve	221
5.21	Histogram of Programming Score in The Post-Test with Normal Curve	224
6.1	The Reasoning Flow for The Programming Process Related Tasks	235
6.2	Programming Process Support Model	237
6.3	Perspectives of Multi Levels Representation of Program	242
6.4	Elements Relationship in The Integrated ALPROV Model	248
6.5	A Cross-Comparison of Difference Tests	257

LIST OF APPENDICES

APPENDIX	TITLE	PAGE
A	Survey Questionnaire	299
B	Syllabus for Programming Fundamental	303
C1	Descriptive Statistics – Respondents from UTeM	307
C2	Descriptive Statistics - Respondents from UDinus	308
C3	Table of Frequencies – Respondents from UTeM	309
C4	Table of Frequencies – Respondents from UDinus	309
C5	Histogram of Trace Score – Respondents from UTeM	310
C6	Histogram of Write Score – Respondents from UTeM	310
C7	Histogram of Trace Score – Respondents from UDinus	311
C8	Histogram of Write Score – Respondents from UDinus	311
D1	Normality Test Using Kolmogorov-Smirnov and Shapiro Wilk	312
D2	Difference Test Between Groups Using Mann-Whitney U Test	313
D3	Difference Test Between Groups Using Kolmogorov-Smirnov Test	314
E	Class Diagram of Main Components in 3De-ALPROV Tool	315
F1	Swimlanes Diagram for Constructing Algorithm Design	319
F2	Swimlanes Diagram for Observing Designed Algorithm	320
F3	Swimlanes Diagram for Identifying Code Mapping	321
F4	Swimlanes Diagram for Executing a Complete Program	322
G1	Sequence Diagram for <i>New Document</i>	323
G2	Sequence Diagram for <i>OpenDocument</i> .	324
G3	Sequence Diagram for <i>Save Document</i> and <i>Print Document</i>	325
G4	Sequence Diagram for <i>Export to C++ Code</i>	326
G5	Sequence Diagram for <i>Export to PNG Image file</i>	327
H	Krejcie and Morgan Table for Determining Sample Size	328
I	Survey Questionnaire for Programming Performance	329
J1	Pre-Test Result of Programming Self Efficacy from Control Group	333
J2	Pre-Test Result of Programming Self Efficacy from Treatment Group	334
K1	Post-Test Result of Programming Self Efficacy from Control Group	335
K2	Post-Test Result of Programming Self Efficacy Treatment Group	336
L1	Paired Sample T-Test from Control Group Between Pre- and Post-Test	337
L2	Paired Sample T-Test from Treatment Group Between Pre- and Post-Test	337
L3	Sample T-Test for Self Efficacy Score Between Control and Treatment Group.	338
M1	Programming Performance Test from Control Group	339

M2	Programming Performance Test from Treatment Group	340
N1	Difference Test Between Control and Treatment Group Based on the Pre-Test Programming Score	341
N2	Difference Test Between Control and Treatment Group Based on the Post-Test Programming Score	344
N3	Difference Test on the Control Group Between Pre-Test Programming and Post-Test Programming Score.	346
N4	Difference Test on the Treatment Group Between Pre-Test Programming and Post-Test Programming Score	349
N5	Independent-Sample Test between Groups Based on the Type of Questions	352
N6	Dependent-Sample Test between Test Sessions Based on the Type of Questions	355
O1	Descriptive Statistic of The Elements of Programming Performance	358
O2	K-Related-Samples Test Between Fundamental, Design, and Development Score	359
P	Expert Evaluation Form and Initial User Interface Design	361
Q	Computer System Usability Questionnaire (CSUQ)	362

LIST OF ABBREVIATIONS AND GLOSSARIES

3De-ALPROV	-	<i>Design-Development-Debug Algorithm-Program Visualization.</i> A prototype of the program comprehension tool that is built based on the integrated model of algorithm-program visualization.
ACM	-	<i>Association for Computing Machinery.</i> The largest and oldest international organization of scientific and educational computer society in the industry.
ALPROV	-	<i>Algorithm-Program Visualization.</i> The new model of visualization tool that emphasizes learning programming on the process of the program and multiple program's abstraction

AV	- <i>Algorithm Visualization</i> . The visualization of the higher-level abstractions which describe software (Stasko et al., 1998).
BLS	- <i>Bureau of Labour Statistics</i> . The principal Federal agency in United States that responsible for measuring labour market activity, working conditions, and price changes in the economy.
CCS	- <i>Computing Classification System</i> . A subject classification system for computing devised by the Association for Computing Machinery (ACM).
CFG	- <i>Control-Flow Graph</i> . A representation, using graph notation, of all paths that might be traversed through a program during its execution.
CS	- <i>Computer Science</i> . A wide range of computing discipline that falls into three categories: design and implement software, devise the new way to use computer, and develop effective ways to solve computing problems.
CSD	- <i>Control Structure Diagram</i> . A notation for the graphical representation of algorithms in detailed designs as well as actual source code. It is intended to reduce the time required to comprehend software by clearly depicting the control constructs and control flow (James H. Cross et al., 1998).
DSV	- <i>Data Structure Visualization</i> . An interactive animation and fairly self-explanatory for understanding a variety of data structures and algorithms, developed by David Galles in University of San Fransisco (Galles, 2011).
DV	- <i>Data Visualization</i> . The presentation of detailed and dynamic views of specified variables, and their structure, within executing code that allows users to observe their behaviour as the execution sequence progresses, providing invaluable debugging information (Ellershaw, Oudshoorn, 1994).
eL-CID	- <i>e-Learning to Communicate Iterative Development</i> . An XML application system to show students examples of computer programs' iterative development using visualisation techniques(Boisvert, 2009)
IDE	- <i>Integrated Development Environment</i> , A software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of a source code editor, build automation tools and a debugger.
IEEE-CS	- <i>Institute of Electrical and Electronics Engineers-Computer Society</i> . The computing professional's single, unmatched source for technology - information, inspiration and collaboration. By making the most up-to-date and advanced information in the computing world
IT	- <i>Information Technology</i> . The technology involved in acquiring, storing ,processing, and distributing information by electronic means, including radio, TV, telephone and computers
ITiCSE	- <i>Innovation and Technology Computer Science Education</i> . Conference from special interest group on computer science education (SIGCSE) that has been held annually since 1996, in many different countries to address pressing issues in computing education.

JDI	- <i>Java Debugger Interface</i> . A high level Java API providing information useful for debuggers and similar systems needing access to the running state of a (usually remote) virtual machine.
JSP	- <i>Jackson Structured Programming</i> . A method for structured programming based on correspondences between data stream structure and program structure.
MCSA	- <i>Multiple-Choices Single-Answer</i> . A question in which respondents are asked to select the best possible answer out of the choices from a list.
PV	- <i>Program Visualization</i> . The visualization of actual program code or data structures in either static or dynamic form (Stasko et al., 1998)
RAPTOR	- <i>Rapid Algorithm Prototyping Tool</i> . A visual programming environment, designed specifically to help students envision their algorithms and avoid syntactic baggage (Carlisle et al., 2005).
SD	- <i>Standard Deviation</i> . A value that shows how much variation or dispersion from the average exists.
SOLO	- <i>Structure of the Observed Learning Outcome</i> . A model of classifying learning outcomes in terms of their complexity, enabling to assess students' work in terms of its quality.
SV	- <i>Software Visualization</i> . The use of the crafts of typography, graphic design, animation, and cinematography with modern human-computer interaction and computer graphics technology to facilitate both the human understanding and effective use of computer software (Stasko et al., 1998).
SWEBOK	- <i>Software Engineering Body of Knowledge</i> . A baseline for the body of knowledge for the field of software engineering that have been established by IEEE Computer Society.
UDiNus	- <i>Universitas Dian Nuswantoro</i>
UTeM	- <i>Universiti Teknikal Malaysia Melaka</i>
VP	- <i>Visual Programming</i> . The use of "visual" techniques to specify or manipulating program elements graphically with visual expression, spatial arrangement and graphic symbols.
WYSIWYC	- <i>What You See Is What You Code</i> . A “radically dynamic” development model to facilitate learner-constructed algorithm visualizations in which the line of algorithm code currently being edited is re-evaluated on every edit, leading to the dynamic update of an accompanying visualization of the algorithm (Hundhausen, Brown, 2007).

LIST OF PUBLICATIONS

Affandy, Suryana, N., & Husin, B. (2014). The Integrated Software Visualization Model to Support Novice's Program Comprehension, *Advanced Science Letters*, 20(12), 2166-2170.

Affandy, Suryana, N., & Husin, B. (2014). Effectiveness of Integrated Algorithm-Program Visualization: A Case Study with the 3De-AIProV. *Advanced Science Letters*, 20(1), 304–308.

Affandy, & Suryana, N. (2012). Integrated Algorithm-Program Visualization: a Novel Approach of Software Visualization Development. In *Mobile Learning, Applications, and Services (mobilcase 2012)*.

Affandy, Suryana, N., Salam, S., & Azmi, M. (2011). The Development of Synergetic Programming Visualization Tool For Novice Programmers, Short-term Project Report, PJP/2010/FTMK(12E)-S720,FTMK-UTeM.

Affandy, Suryana, N., Salam, S., & Azmi, M. (2011). 3De-synergetic Program Visualization: A visual learning-aid tool for novice students. In *e-Education, Entertainment and e-Management (ICEEE), 2011 International Conference on* (pp. 133–137).

Affandy, Herman, N. S., Salam, S., & Noersasongko, E. (2011). A Study of Tracing and Writing Performance of Novice Students in Introductory Programming. In J. M. Zain, W. M. bt Wan Mohd, & E. El-Qawasmeh (Eds.), *Software Engineering and Computer Systems* (pp. 557–570). Springer-Verlag Berlin Heidelberg.